International QL Report

November/December 1991

Published by SeaCoast Services

Published 4 - 6 times per year | Volume I, Number 4 | $3.00

EDITORIAL

NEWPORT, RHODE ISLAND, USA - THE EDITORIAL STAFF

The holiday season will soon be upon us, and I'm sure most of us have a lot to be thankful for. We would like to wish you and yours a happy, healthy, and prosperous holiday season.

With this issue we have reached the milestone of issue #4, and are collecting articles and tidbits for issue #5. In recent conversations with subscribers, we found a widespread belief that we had all the material we wanted, so they had not sent anything in for publication. Nothing could be further from the truth. Folks, we really welcome your input. Many European readers have asked for more articles by North Americans. As we enjoy reading about what is going on in the rest of the QL world, they are equally interested in what's going on here. So, please, send in your articles, listings, bits of information, etc. The only restriction on the number of pages we print is the amount of material on hand.

Many of you have indicated that you would have loved to have been included in the GOLD CARD buy. We are happy to inform you that the good people of Miracle Systems, are willing to extend the 27% savings we received, for a limited period (great Christmas gift). For more information and ordering instructions, call Bob at IQLR (401-849-3805).

The "IQLR BUYER'S REGISTRY" (see issue #3) have placed their first group buy. The item ordered is the Keyboard-90 Interface from the manufacturer at a 25% discount. If you haven't joined the BUYER'S REGISTERY, this might be a good time.

We are constantly looking for ways to improve and enhance IQLR, to this end, our USA readers will now receive their issues in envelopes. This will free up a half page for additional material, and hopefully ensure it's arrival in good condition. Let us know what changes or additions you'd like to see in the content of IQLR.

CORRECTION

Issue #1 of IQLR contained a program listing for DIRECTORY-DEVICES. Unfortunately the listing acquired an error when I typed in a "+" instead of a "=" (both symbols are on the same key). I apologize for any inconvience or frustration this may have caused. This is a useful little program and the author was in no way responsible for the error. The following two lines are what should have been listed:

```
300 REMark if pointer=0 then end of list
310 IF ad=0 THEN EXIT loop
```

3.2 MEG DISK DRIVES

Miracle Systems has informed us that they have an upgrade ROM (version 2.24) for the GOLD CARD that they will provide free of charge when requested by individuals who purchased earlier versions. This replacement chip improves the use of the 3.2 meg drives as well as providing a "slow down" option that will make arcade type games more playable.

Miracle Systems has put a package together that takes the guess work out of buying the right drives. They are offering two 3.2 meg drives, cased with power supply (you must tell them that you need a power supply rated 110/120 volts for service in the USA), they also include a box of 10 Extra High Density disks. The cost of this package is 155 Pounds Sterling (this is a better price than what we can buy the items separately for).

Miracle Systems Ltd.
25 Broughton Way
Osbaldwich, York YO1 3BG
U.K. Tel: 0904 423986

They accept VISA and MASTERCARD. From the US dial: 011-44-904-423986

IN THE PIPELINE

Work continues on the GRAPHICS CARD from Miracle Systems, as well as TWO NEW PRODUCTS. The first is an honest to goodness, PARALLEL PORT, and the second is a true SERIAL PORT. While no delivery dates were given by Miracle on these last two items, they are expected in the not too distant future.

All three of these items, when released, will further open up the QL to the outside world, as well as support the development of a new SUPER QL.

REPLACE: AN ARCHIVE UTILITY

CORNISH, NEW HAMPSHIRE, USA - BILL CABLE

Editors's note: The listing that accompanies this article was printed on a 24 pin dot matrix printer directly from the DOC file received on disk from Bill . Hopefully this will prevent typos and avoid the frustration that comes from keying in a long listing only to find it won't run. Database systems and database oriented software play an increasingly important role in personal computing. Computers are fantastic for sorting and using information, tasks that database systems are specifically designed for. When you purchased your QL you received a well designed and powerful database system in ARCHIVE.

Known more as a developer's tool, ARCHIVE has made a negative impression on many QL'ers. This is unfortunate because database oriented software can only come from the ARCHIVE language, or other specialized programs written in non-database languages like "C" or SuperBasic.
It takes enormous effort to enter large amounts of data into a database, and you want to get the most out of it. Any application program that builds up a body of data needs to support very specialized data manipulation and reporting through it's own internal language, or else

interface with another database system that does. Without the fore-mentioned flexibility you would be unable to properly use or control your data. For us, ARCHIVE provides this complete data handling environment, interfacing with most good QL database oriented software through it's export/import functions. In many ways we're lucky to have this common ground, rather than the fragmentation seen on the PC.

Why REPLACE?? Often the information in a large database contains inconsistencies which interfere with useful structuring. For example, you have an address database with hundreds of addresses from around the world and you want to organize it by country. Unless you were extremely organized in your up-front planning, you probably weren't consistent when you entered the country name, for example, UK, U.K., England, ENGLAND, GB, USA, U.S.A., or United States. If there were a lot of such inconsistencies, it'd take a considerable amount of time to find and alter the data. With REPLACE it's quick and easy to change occurrences like England to UK one time or 1000 times. If you use large databases, you'll certainly be able to find many uses for this utility.

REPLACE allows you to replace one piece of text with another in any ARCHIVE database, you can do it by a yes/no query or repeatedly (all) or repeatedly with suppressed display (fast). You can do it in one field or in all fields, upper and lower case can be ignored or not. With this in mind, REPLACE should be part of your bag of tricks when using ARCHIVE.

REPLACE is a complex program that does a lot of text manipulation. I've arranged the listing for both compactness and completeness of each line. It's not easy to follow in such a form, but you should be able to pick out the main points. Incorporated within the program are many useful general purpose procedures that I've developed over the years for standardized input and output.

This is a NEW program and I've debugged as much as I can, but be watchful, and please report any bugs you may find to me. Never use it on your master copy of a database, but only on a backup. When you're satisfied with the results, then and only then should you overwrite your master. Any ordering of your database will be lost, so remember to reorder it when you're done.

I am placing this utility in the public domain, and it will be available from various group libraries. An enhanced version plus many other useful ARCHIVE utilities are available from me as "DBProgs", the price is $19.95. Send orders to:

Bill Cable
RR 3, Box 92
Cornish, NH 03745
USA Tel: 603-675-6081

To enter the program; boot up ARCHIVE, type: **EDIT <ENTER>**, then type in the listing, procedure by procedure. The ARCHIVE editor will automatically indent just like the listing. If your indentations don't match the listing then you have made a mistake. EXIT the editor to SAVE (**SAVE** command) or print (**LLIST** command) the program. To start the program type: **replace <ENTER>**. Try it on the GAZET database (the sample database that came on the ARCHIVE cartridge), by changing **"$"** to **"DOLLAR"**, etc.

HAPPY ARCHIVING, Bill Cable

(REPLACE: listing)

```
proc acky;line,m$,v$
  liny;line,v$: print v$;m$;". Press <ENTER> : ";v$;: input i$:liny;line,""
  endproc

proc close_all
  while 1: close : endwhile
  endproc

proc defy;line,m$,d$,v$
  liny;line,v$,"": print v$;m$;" [<ENTER> if '";d$;"'] : ";:inp
  if ans$="": let ans$=d$: endif : if ans$="\": let ans$="": endif
  print at line,len(m$)+1;" : ";rv$;" ";ans$;" ";rv$; tab 80;v$;
  endproc

proc displayer
  let i$=fl$: let i=instr(i$,","): print at 3,0;: while i
    let fn=int(val(i$(1 to i)))
    if fn>=0 and fn<numfld()
      print fieldn(fn); tab 13;": ";fieldv(fn); tab 79: endif
    if i<len(i$): let i$=i$(i+1 to len(i$)): let i=instr(i$,",")
      else : let i=0: endif : endwhile
  endproc

proc field_find;k
  let fnd=0: let fld=numfld(): while k<numfld(): if fieldt(k)
      let i$=fieldv(k): if ign: let i$=lower(i$): endif
      if instr(i$,s$): let fnd=1: let fld=k: let fld$=fieldn(k): return
        endif : endif : let k=k+1: endwhile
  endproc

proc field_show
  let pg=1: cls
  print "Fields of ";dv$;db$; tab 40;count();" records"; tab 70;"page ";pg
  let i=0: let r=2: let c=1: while i<numfld(): let i$=num(i,3)
    print at r,wid*(c-1);i$;" ";fieldn(i);: let i=i+1: let r=r+1
    if r>rows+1: let c=c+1: let r=2: if c>cols and i<numfld()
        input at 23,1;"Press <ENTER> for more : ";i$: let c=1
        let pg=pg+1: cls : print "Fields of ";dv$;db$; tab 70;"page ";pg
        endif : endif : endwhile
  endproc

proc header;i$
  paper spap: cls : paper hpap: ink hink: print rept(" ",160);
  print at 0,1; tab (80-len(i$))/2;upper(i$); at 3,0;: paper spap: ink sink
  endproc

proc inp
  print rv$;: input " ";ans$;" ";: print rv$;
  endproc
```

(REPLACE: listing cont.)

```
proc inpy;line,m$,v$
  liny;line,v$: print v$;m$+" : ";:inp: print v$;
  endproc

proc key_choice;i,i$,j$,k$
  let ans$="": while not instr(j$,ans$) or len(ans$)<>1
    print at i,0;k$; tab 80; at i,1;i$;" : ";rv$;" ";rv$;k$;
    let ans$=lower(getkey()): if not instr(j$,ans$)
      acky;i,"Press the SINGLE KEY corresponding to the desired action",rv$
      endif : endwhile : print k$;rv$;ans$;" ";rv$; tab 80;k$;
  endproc

proc liny;line,v$
  if line
    print at line,0;v$; tab 80;v$;: print at line,1;: else : print : endif
  endproc

proc make_proc
  if lt$<>fld$: if kc$<>"f":msg;23,"making transfer procedure","": endif
    spoolon dv$+"tmp_tmp" export : lprint "proc transfer"
    lprint "let ";fld$;"=j$": lprint "let lt$='";fld$;"'"
    lprint "endproc": spooloff : merge dv$+"tmp_tmp": endif
  endproc

proc msg;line,m$,v$
  liny;line,v$: print v$;"{"+m$+"}";v$;
  endproc

proc replace
  error close_all
  mode 0: let ign=0: let hpap=5: let hink=0: let spap=0: let sink=5: let lt$=""
  let rv$=chr(0)+chr(26): let rows=20: let cols=4: let wid=80/cols: let kc$=""
  header:"replace procedure for archive databases"
  print at 2,38;"V1.0 For Public Domain   Bill Cable  9/91"; at 4,0;
  print "Replaces text within any database in one or all text fields. You can"
  print " make changes that cannot be reversed so work with a backup copy of"
  print " your database not with your master copy. Works best using RAM files."
  defy;8,"Device where database is (ram1_,flp1_,mdv1_,etc)","ram1_",""
  let dv$=ans$: dir dv$:inpy;23,"Database name [ Not your master copy ]",""
  if ans$="": mode 1: stop : endif : let db$=ans$: open dv$+db$
  msg;23,"removing any order","": reset :field_show
  print at 22,0;"Below separated by commas list field numbers for display"
  defy;23,"Field numbers to display","0","": let fl$=ans$+","
  while 1:header;"replace text in "+dv$+db$
    print "Give text to search for [<ENTER>=exit]"
    print at 5,1;"Search for ";rv$;" ";: input s$;: print " ";rv$;
    if s$="": close : mode 1:msg;0,"All files closed","": stop : endif
    print " and replace with ";rv$;" ";: input t$;: print " ";rv$;
    yorn;7,"Do replacements in one particular field",""
    if ans$="y":field_show: let t=0
```

(REPLACE: listing cont.)

```
    while not t:inpy;23,"Replace '"+s$+"' with '"+t$+"' in field number",""
      let fld=int(val(ans$)): if fld>=0 and fld<numfld(): let t=fieldt(fld)
        endif : endwhile : let fnd=1: let fld$=fieldn(fld)
    let m$=fld$: let n$="search "+fld$+" field":make_proc
    else : let m$="all": let n$="search all fields": let fnd=0: endif
  yorn;23,"Ignore upper/lower case in search",""
  if ans$="y": let ign=1: let s$=lower(s$): else : let ign=0: endif
  yorn;23,"Ready to "+n$+" to replace '"+s$+"' with '"+t$+"'",""
  if ans$="n": let kc$="e": else : let kc$="": let fcnt=0: let rcnt=0
    header;"DOING REPLACEMENTS":msg;23,n$+" for '"+s$+"'",""
    if m$="all": find s$: else : if not ign: search instr(fieldv(fld),s$)
       else : search instr(lower(fieldv(fld)),s$): endif : endif : endif
  while found() and kc$<>"e"
    if m$="all": if not fnd:field_find;0: endif : endif
    if fnd or m$<>"all": if kc$<>"f":displayer: endif : let fldv$=fieldv(fld)
      if ign: let k=instr(lower(fldv$),s$): else : let k=instr(fldv$,s$)
        endif : let chg=0: let l=len(s$): while k and kc$<>"e"
        let l$=fldv$(k to k+l-1): if k=1: let k$=rv$+l$+rv$
          if l<len(fldv$): let k$=k$+fldv$(l+1 to len(fldv$)): endif
          else : let k$=fldv$(1 to k-1)+rv$+l$+rv$: if k+l-1<len(fldv$)
            let k$=k$+fldv$(k+l to len(fldv$)): endif : endif
        print at 20,1;fld$; tab 13;": ";k$; tab 79: let fcnt=fcnt+1
        if kc$<>"a" and kc$<>"f": let i$="Replace Options : "
          key_choice;23,i$+"Y(es)/N(o)/A(ll)/F(ast)/E(xit)","ynafe",""
          let kc$=ans$: if kc$="f":msg;23,"in fast replace mode",""
            else : if kc$="a":msg;23,"replacing found text","": endif : endif
          endif : if kc$="y" or kc$="a" or kc$="f"
          let rcnt=rcnt+1: if m$="all":make_proc: endif : if k=1: let j$=t$
            if l<len(fldv$): let j$=j$+fldv$(l+1 to len(fldv$)): endif
            else : let j$=fldv$(1 to k-1)+t$: if k+l-1<len(fldv$)
              let j$=j$+fldv$(k+l to len(fldv$)): endif : endif
          print at 20,15;j$; tab 79;: let fldv$=j$: let k=k+len(t$): let chg=1
          else : let k=k+l: endif
        if k<=len(fldv$): if not ign: let i=instr(fldv$(k to len(fldv$)),s$)
           else : let i=instr(lower(fldv$(k to len(fldv$))),s$): endif
          if i: let k=k+i-1: else : let k=0: endif : else : let k=0: endif
        if kc$<>"f":msg;23,"more "+n$+" for '"+s$+"'",""
          endif : continue : endif : endif : endwhile
  if kc$<>"e":acky;23,"Reached end of file",rv$: endif : endwhile
endproc

proc yorn;line,m$,v$
  let ans$="": while ans$<>"y" and ans$<>"n":liny;line,v$
    print v$;m$+" [ y/n ] : ";:inp: let ans$=lower(ans$): print v$;: endwhile
  endproc
```

Editor's Note: Bill has produced a wide range of ARCHIVE software products and educational programs and is a regular contributer to UPDATE Magazine on the subject of ARCHIVE. Their back issues contain a treasure trove of Bill's material.

UPDATE MAGAZINE supports the 2068, Spectrum, and QL computers, it is published quarterly at the subscription rate of $18.00 per year. Why not subscribe now and enjoy Bill's articles year round.

Write to: UPDATE MAGAZINE, P.O. Box 1095 - Peru, IN 46970, USA

ONLINE WITH THE QL

IDAHO FALLS, IDAHO, USA - RON BLIZZARD

The September/October IQLR mentioned that 41% of those responding to the questionnaire used their QL's primarily for communications. Although, this number is larger than expected, it still indicates that over half of all QL owners DO NOT use modems, even though it is now cheaper than ever to get "online".

Why use a modem? There are many reasons. You can join one of the information services (like Genie, Delphi, or Compuserve) and tap directly into the AP newswires, or join a Special Interest Group (SIG), or send and receive electronic mail (E-Mail), or even play online games. You can buy stocks by phone and, in some places, even do your banking via modem.

Or maybe you want to communicate with other computer users in your area. There are thousands of Electronic Bulletin Boards (BBS) that specialize in hundreds of subjects. Several, including GREYMATTER (213-871-6260) in Los Angeles and C.A.T.S. BBS (301-588-0579) in Washington, D.C., cover the Sinclair computers and have public domain or shareware programs that can be "downloaded" to your QL. If you have a problem, there is usually someone available who has had a similar problem and willing to help.

Then again you may just be curious, or want another challenge, or need to find a use for your SER2 port. That's how I got involved with communications. I wanted to see what was out there and to use my QL for something new.

One of my first discoveries was Nuclear Fiction, an "online" Science Fiction magazine that later published my first two short stories. This taught me a lot about writing because we were encouraged to critique each other's work. I also made friends and, after bragging about the QL, found that one of the writers that I had been critiquing, and having discussions with, owned a QL and was a former president of the Dallas T/S User's Group (It's a small world).

The main benefit I've received from using a modem, however, is the QL information I've collected. My disk I/F and drives were bought through online advertisements. The solution to my Expanderram/Delta Disk I/F compatibility problem came over the phone lines. I've gotten software advice, found sources of QL news, and found out about special services, all via modem. And I've made a lot of friends in the process.

There are drawbacks. Long distance phone calls can get expensive (thankfully Ed Grey, of GREYMATTER, let me know about PC-Pursuit -- an online service that let's you call BBSs in all major cities in the evenings for a flat monthly fee -- or I'd still be paying AT&T). There are fewer Sinclair specific BBSs now.and many of the online programs have gotton "long in the tooth", but there are still some useful ones. For example, READMAC, a program that allows you to transfer MacPaint files to the QL, is available. This means that online graphic files being created for the Macintosh can be used in QL DTP and paint programs. There are similar programs for transferring .GIF and .RLE files to QL format.

All this and more is available for the cost of an external modem and the parts to make a cable. You'll also need terminal software, but there are two very good public domain packages available --QLINK and QL-52. QLINK comes with a built-in editor, and several utilities, including filters for text files. Both programs can use XMODEM for downloading and uploading files to BBSs or to friends with computers. QLINK does not need the MODAPTER at 1200 baud, though, I'm told, it helps. I have a MODAPTER PLUS and it is incompatible with QLINK.

The following cable diagram works fine with my Avatex 1200 baud modem and QLINK (taken from the QLINK manual).

QL (SER2ir) (DB-9)	Modem (DB-25)
1- GND ---------->	7- Signal Ground
2- TxD ---------->	2- TD
3- RxD ---------->	3- RD
4- No Connection	
5- No Connection	
6,7,8 GND (The shielding ground in the shielded cable --connected to three pins on the QL side)	
9- +12v ---------->	20- DTR
9- +12v ---------->	4- (two wires are connected to the same pin on the QL side)

A 1200 baud external modem (no longer state of the art) is available for $19.99 plus $5.50 S/H from Damark. A 2400 baud external modem can be found for about $75. I can supply a copy of QLINK or QL-52 if you send either a floppy disk (5 1/4" - 1440 sectors) or a formatted microdrive and return postage. I would also be glad to answer any questions you have -- provided I know the answer.

Ron Blizzard
6479 East 97 North
Idaho Falls, ID 83401
(208) 523-2330

Damark
1-800-729-9000
Atari Model #SX-212
Item # B-376-181504

TOOLKIT II TUTORIAL - PART 3

ADAPTED FROM: QL TECHNICAL REVIEW (C.G.H. SERVICES)

6. SUPERBASIC PROGRAMS

6.1 DO

DO is a command for an executed SuperBasic command file, which is a file containing unnumbered BASIC statements. Thus, using the example from 5.5.3, the command: DO PRINT_CMD, would perform the three spooler commands contained within the file. The advantage of the DO command being that the current SuperBasic program is unaffected. It would be lost if you used LRUN. Any block commands within a command file must appear on a single line, for example:

```
FOR n = 1 TO 10:  PRINT n
REPeat read: INPUT a$: PRINT a$, CODE (a$)
```

would be an acceptable file, whereas the following would not:

```
FOR n = 1 TO 10
   PRINT n
END FOR n
REPEAT read
   INPUT a$
   PRINT a$, CODE (a$)
END REPeat read
```

An attempt to LRUN such a file would lead to the error "not found". This refers to loop control 'a' which only exists in the line of the definition. It is of course, acceptable to use either upper or lower case for keywords, and use normal abbreviations. Note the warnings at the end of 6.1 in the TKII manual.

6.2 DEFAULT DIRECTORIES

The normal BASIC filing commands have been modified to use the default directories. In addition the LOAD command will look for a file in the PROGRAM default, if it doesn't locate it in the DATA default directory. An overwrite variant of the SAVE command, SAVE_O has been introduced that works in the same manner as other overwrite commands.

7. LOAD AND SAVE

This section refers to the loading and saving of binary files, i.e. LBYTES and SBYTES for resident procedures and EXEC, EXEC_W, and SEXEC for transient programs. SBYTES and SEXEC have been modified in the same way as other commands that write to files (prompt appears if file already exists), and the overwrite variants have been introduced.

A new command, LRESPR, has been added that combines the functions of RESPR, LBYTES, and CALL. Thus: base = RESPR (file_length): LBYTES file, base: CALL base, can simply be performed by typing: LRESPR file. With the latter it's not necessary to explicitly find out the length of the file. As with RESPR, LRESPR may only be used if no other jobs, other than BASIC are running on the QL.

8. PROGRAM EXECUTION

This section deals with the commands for executing compiled programs which run on the QL as jobs. This formerly consisted of two commands EXEC and EXEC_W. These have been modified and made synonymous with new versions: EX and EW. Another command, ET has been introduced which loads a program into memory but returns control to BASIC before starting the job. The EX command is explained further to illustrate the new facilities provided by all of these commands.

8.1 SINGLE PROGRAM EXECUTION

EX may be used in the same way as the standard EXEC command in order to start a job on the computer: EX filename. The command will look for the file on the program default directory. In addition, the program may be passed a parameter string. As an example of use I'll refer to a commercial program 'MASTER SPY EDITOR'. This program can be invoked as follows: EX MS, FLP1_BOOT. This command executes MASTER SPY (which I've renamed to MS on my working copy) which loads the file FLP1_boot and presents it ready for editing. This feature was made available on MASTER SPY (version 1.7 and onwards) as a result of my writing to ARK to ask if it were available.

A further feature of the EX command is that filename (or channels) may be passed to a program for use as it's standard input and output. BASIC programs compiled using SUPERCHARGE cannot be passed input and output files, perhaps TURBOCHARGED programs can, I don't know. But it is easy to write a PASCAL program to accept filenames for input and output channels, and is a standard feature of PASCAL. Below is an example PASCAL program which should be quite easy to follow for anyone familiar with SuperBasic. Comments are enclosed between curly brackets.

```
PROGRAM mul2 (input, output);
VAR
   param : string [20];              {like DIM param$ (20)      }
   in_num, out_num : real;

BEGIN
   REPEAT
      getcomm (param);               {read the parameter string  }
      writeln  (param)
      readln  (in_num)               {equivalent to INPUT a      }
      out_num := in_num * 2
      writeln (2 * in_num);          {equivalent to print 2 * a  }
   UNTIL in_num = 0;
END.
```

This program simply reads in numbers and writes out double the number. If the program was invoked using EX MUL2_BIN (the file mul2_pas is passed to the compiler which produces mul2_rel [the extension .obj would be used on MS-DOS systems] and then the linker processes this file and produces mul2_bin [.exe under MS-DOS]) then the numbers could be typed in at the keyboard, and the answers would be printed to the screen. Because no parameter has been passed only the numbers would be displayed on screen, however the same program could be invoked as follows:

```
EX MUL2_BIN, IN_DAT< OUT_DAT; 'IN_DAT * 2'
```

MUL2 would have to be located in the program default and IN_FILE in the data default directory. If IN_FILE contained the following lines:

```
2.7
-3.34
10.6
0
```

Then the file OUT_DATA would be produced in the data default directory containing the following lines:

```
 IN_DAT * 2
  5.4000000E+00
-6.6800000E+00
  2.1200000E+01
  0.0000000E+00
```

The file OUT_DAT would be overwritten automatically if it already exists. Note the numbers may be easily formatted so as not to use scientific notation, this is merely the default. The same results could be achieved by passing channel numbers instead of file names:

```
OPEN_IN #3, IN_DAT
OPEN_NEW #4, OUT_DAT
EX MUL2, #3, #4; 'IN_DAT * 2'
CLOSE #3: CLOSE #4
```

The Propero PASCAL compiler and the GST LINKER also accept parameter strings. The compiler uses the parameter string to pass the name of the PASCAL program and flags indicating various options for the compilation. Likewise with the linker one passes the program name and the name of the file containing the linker directives. The parameter need not be a string constant, it could be a variable:

```
FILE$ = FLP1_BOOT : EX MS;FILE$
```

8.2 FILTERS

EX also allows a series of programs to be executed that work together to process a stream of data, the output from one program being passed to the input of the next. The situation is analogous to a production line. In the TKII manual it explains that a series of programs (or filters) could be executed as follows:

```
EX UC, FRED, TO LNO TO PAGE, SER; 'FILE FRED' & DATE$
```

Such a series of programs could be easily written in PASCAL but the string handling is sufficiently different from SuperBasic so as to make the example of little use. Instead, consider a simpler set of programs:

```
EX ADD3_BIN, IN_DAT TO MUL2_BIN, OUT_DAT; 'NUMBERS'
```

MUL2 is the same program as listed above. The output from ADD3 goes to the input of MUL2 and the output goes to the file OUT_DAT. The file OUT_DAT will have the heading 'NUMBERS'. The program ADD3 is as simple as MUL2:

```
PROGRAM ADD3 (input, output);
VAR
   in_num, out_num : real;
BEGIN
   REPEAT
        readln (in_num);
        out_num := in_num + 3;
        writen (out_num);
   UNTIL in_num = -3;
END.
```

This program reads a series of numbers and writes the values plus three. It stops when it reads the number -3, this will have three added and be passed to MUL2 which stops when it reads the number 0. So they stop properly together. If any program in the chain failed, then the whole series of jobs involved would be removed.

Suppose that IN_DAT now contains the following lines:

```
2.3
-3.6
10
-3
```

The ADD3 (the job name is derived from the name on the PROGRAM statement in the PASCAL program) will read this file and pass the following numbers to MUL2:

```
 5.3000000E+01
-6.0000000E+01
 1.3000000E+01
 0.0000000E+00
```

MUL2 will read the numbers, and produce the file OUT_DAT:

```
 NUMBERS
 1.0600000E+01
-1.2000000E+00
 2.6000000E+01
 0.0000000E+00
```

The means of communications between these two programs is via a pipe. If the IN_DAT is a much bigger file say, a thousand lines, then while these programs are executing inspection of the channels menu in QRAM shows that there is a pipe associated with both of the programs.

Each of the programs in the chain may have many other channels open and use the screen and keyboard as well as other files and devices. However, if using software like QRAM, it is important to remember that if the programs in the chain are competing for the screen, then one will be suspended, this will cause the chain of programs to fail (the first

program in the chain may be suspended, and this will suspend the chain of jobs once the pipes have been emptied). With the PASCAL programs as described, the programs will fail even though output is not sent to the screen. This situation may be remedied by using the UNLOCK utility supplied with QRAM.

I would think that it would be possible to write similar programs in FORTRAN, in which case unit 6 of one program would be attached via a pipe to unit 5 of the next. 'C' also has standard input and output, which I'm sure would accept pipes (on a full implementation of the language).

END OF PART 3
AUTHOR: Stephen Bedford

TIDBITS

ST. THOMAS, US VIRGIN ISLANDS - TAYLOR S. PENROSE

The following short routines may be useful to you:

LISTING 1

```
100 DEF PROC Frame (Txt$)
110 Cx$="  ": LineLen=52: REM Cx$=Space: Change LineLen to Suit
120 FOR Cx=LineLen to 1 STEP -1: REM Move Backwards
130 IF Len(Txt$)<LineLen: EXIT Cx
140 IF Txt$ (Cx)=Cx$ THEN CC=Cx: PRINT Txt$(1 TO CC)\:
    Txt$=Txt$(CC+1 TO ): REM look for space to break line
150 END FOR Cx
160 PRINT Txt$
170 END DEF: REM neatens text: what width should do!
```

LISTING 2

```
100 DEF FN Sign(Num)
110 IF Num<0: RET-1
120 IF Num>0: RET+1: ELSE RETurn 0
130 END DEF: REM A feature of older Basics
```

REFLECTIONS

What kind of year has it been for the QL? As an individual involved with the QL since it's introduction in 1984, I can honestly say, that the level of co-operation and development has reached heights never before attained. I'd like to review some of the hardware and software developments, both commercial and public domain, an important new group, a couple of new publications.

HARDWARE: The following items are by no means all inclusive of this past year's offerings, but those that have had a major impact:

The Miracle GOLD CARD
The KEYBOARD-90 INTERFACE
The MINERVA MK II ROM
The (resurrected) QIMI MOUSE INTERFACE
The QL ROM-CARD with MOS DISK DRIVER

(Note: Refer to back issues of IQLR for detailed information on the above listed items). When you add to this list the rumored GRAFIX + CARD from Miracle, and a low-cost simple HARDDISK interface that may be offered by QUANTA. You can readily see that the hardware development arena is humming with activity.

SOFTWARE: This year's High Quality professional software releases include:

DP's PERFECTION, word processor
Jochen Merz's QD III, Mouse driven Editor
Jochen Merz's QDESIGN, Mouse driven CAD/Drawing package
The Prog's DATADESIGN, Mouse driven Database
Dilwyn Jones' PICTUREMASTER & VISION MIXER PLUS
C.G.H. Services' QL WORLD D-I-Y TOOLKIT

The availability of Public Domain software has grown a hundred fold with C.G.H. Services, and Qubbesoft, both of the UK, and Intergroup Freeware Exchange in Germany taking the lead. You probably expected QUANTA in this group, but unfortunately they haven't added anything new to the QUANTA Library in over a year. A couple of major public domain releases during this last year includes:

The C-68 'C' Compiler (7 disks)
MINIX-QL

A SPECTRUM EMULATOR is anticipated for release into the public domain (of interest to Spectrum and emulated 2068 owners), before the end of the year.

There are a number of other software and hardware projects in the works, and when we verify that they are more than just pipedreams, will keep you informed.

PUBLICATIONS & SPECIAL INTEREST GROUPS: In North America two new publications have been launched this year. The first is the "QL HACKER'S JOURNAL" that provides a forum for QL programmers to exchange ideas and programs. The second is the "IQLR" you are reading.

A special interest group, QLAW headquartered in the UK was formed to promote a second generation QL. They are trying to bring the people, expertise, and money, together so that a new Super QL will no longer be a dream but a reality. If you haven't sent in the questionaire that appeared in issue #1 of IQLR and shortly thereafter in QL World, PLEASE do so, as the information requested is crucial to their success. You don't have to join QLAW, just fill out and send in the questionaire.

SUPER QL - THE NEXT GENERATION

NEWPORT, RHODE ISLAND, USA - BOB DYL

PROBLEMS: In the past there were two major problems restricting the development of the QL. The first ,was Amstrad's copywrite of the QDOS operating system, and their unwillingness to license it to another party. The second major obstacle was the lack of a proper bus system. It was widely reported several years ago that this is what prevented the launch of the Futura.

SOLUTIONS: Both of these problems have now been addressed, at least in my eyes. We have two Legal QDOS-Like operating systems free from Amstrad copywrite. The first and best known is the MINERVA ROM developed by QVIEW. The second is the software portion of the QL emulator for the Atari ST, or the SMS-2 operating system developed by Tony Tebby, the creator of QDOS. Either of these or even a marriage of the two could supply the new QL with a superb QDOS like operating system, that would offer compatibility with existing QL software.

In like manner, the bus problem can be answered by the VME Bus System that is quickly winning acceptance as the standard bus system for industrial computers using the 68XXX family of processors. Miracle in using the 68000 processor in the GOLD CARD has opened the door for development along these lines. There is a large number of engineers working to future proof the VME Bus. A considerable amount of hardware and software already exists for this Bus System.

In discussing this with a number of engineers, several of whom are familiar with the QL, the excitement over the prospect of a new Super QL, with a QDOS Like operating System, and a VME Bus was overwhelming, especially when you consider the possibility of using OS-9 and eight, sixteen and thirty-two bit expansion slots on the board.

Please don't misunderstand, designing any piece of hardware will be plagued with problems that have to be solved, we have just looked at two major ones. A considerable amount of time, energy and money will be required before we see a Super QL.

POSSIBLE PATH SCENARIO: The most logical approach would seem to be the creation of a new motherboard. This would contain the new operating system and VME bus system. Included on this board would be Centronics Parallel and true RS-232 Serial interfaces, as well as 8,16 and 32 bit expansion slots. A proper case, keyboard, and power supply would also be supplied.

At this point the producers of the motherboard could offer it to QL owners to add expansion boards and devices they already own, allowing us to customize our systems based upon need and financial resources (the new motherboard would have to be compatable with all or most of present QL hardware for this to be effective).

The advantages to QL owners are obvious. The advantage to the producers of the new motherboard and case include a stable immediate market, profits that could be used to market a complete NEW SUPER QL to the general computing public (new users), without the enormous outlay of cash that is normally required to launch a new computer. The fact that thousands of software applications (commercial and public domain), already exist would be a tremendous boost to marketability.

The 68XXX family of computers already have hardware/software emulators for most of the worlds computers i.e. MAC, MS-DOS, OS-9, AMIGA and ATARI-ST. The possibilities are endless, and we would be back on the cutting edge with room for expansion.

We have seen that the GOLD CARD prevented some users from abandoning the QL and brought previous users back. We believe the GRAFIX + CARD will do the same. A SUPER QL without a doubt, would swell our ranks. We live in exciting times.

LINKING ASSEMBLER INTO C PROGRAMS

EAST PROVIDENCE, RHODE ISLAND, USA - WILL HORTON

With the advent of the C-68 "C" Compiler (available from IQLR), an entire library of QDOS functions are available. However, there may be times when you need a special purpose routine that isn't available in this library, or if you use the Metacomco QLC which only comes with an all purpose QDOS trap function.

Being able to fully utilize QDOS makes a programming language useful for code generation on the QL. This feature is offered by the "C" language, with the power of assembly and the flexibility of a high level programming language. To further enhance this, you can link your own assembly language routines into "C" programs. This article purports to demonstrate how this is done.

Please refer to the C-program listing below. Note that this program is written for the C-68 compiler.

C-PROGRAM LISTING:

```
#include "stdio.h"
#define chan fgetchid (stdout)

main ()
{

      char dir_name[10];
      int chid,sectors,open_sectors,total_sectors;
      int mask = 0x0000FFFF;

      chid = open_dir ("FLP1_");

      /* read sector information and medium from floppy 1 */
      sectors = medinfo(chid,dir_name);

      /* separate sector information from long word sectors */
      open_sectors = sectors >> 16;
      total_sectors = sectors & mask;

      sd_clear (chan,-1);

      printf("open/total sectors = %d/%d\n",open_sectors,total_sectors);
```

```
        strncpy (dir_name,dir_name,10);
        dir_name[10]=0;
        printf("Device name = %s",dir_name);

        io_close(chid);

}
```

The purpose of this program is to open "floppy 1", read the medium information from the floppy, and print the data to the screen. The medium information being the free sectors, total sectors, and the device name.

The above listing containing the function medinfo() is the assembly language module that will be linked. This function contains two arguments: the channel id, (an integer), and the pointer to the name of the device. This function will return an integer containing the number of sectors on the floppy. The most significant word of this integer is the open sectors, and the least significant word is the total sectors. It will also return the pointer "dir_name" pointing to the name of the floppy.

Now please refer to the Assembly language listing below, this listing is the function medinfo().

ASSEMBLY LANGUAGE LISTING:

```
*
*This program gets information about the medium: the name, and sectors.
*
*               medinfo(chid,dir_name);
*
FS.MDINF        EQU     $45
IO.FSTRG        EQU     $03
*
                XDEF    medinfo
medinfo         MOVE.L  4(A7),A0        drive id
                MOVE.L  8(A7),A1        pointer to heap
                MOVEM.L D2-D7/A2-A5,-(A7)  save registers
                MOVE.L  A1,A4
*
                SUB.L   D5,D5
                MOVEQ   #FS.MDINF,D0
                MOVE.L  (A6),A1
                TRAP    #4              make it absolute
                MOVEQ   #-1,D3
                TRAP    #3
                MOVE.L  D1,D5           store sector info
*
                MOVE.W  #10,D2          name length
                MOVE.L  (A6),A1
AGAIN           MOVE.B  ()(A6,A1.L),(A4)+  load name onto pointer
                ADDA.L  #1,A1
                SUBI.W  #1,D2
```

```
                CMPI.W      #0,D2
                BPL         AGAIN
                MOVE.B      #0,(A4)
*
                MOVE.L      D5,D0               return sector info
                MOVEM.L     (A7)+,D2-D7/A2-A5   restore registers
                RTS                             and quit
*
*
                END
```

The first item to take note of is the assembler directive XDEF. This directive tells the assembler that the name medinfo is an external definition and that it can be called by an external program. In this case it will be called by the C-Program listed in the beginning of this article. Next, two items are popped off the stack, these two items are the arguments of the function medinfo(), chid and dir_name respectively. Now the registers are saved on the stack with a multiple move command. Saving all of the registers (except A6 and A7 which are never saved),is somewhat of a formality, but it won't hurt to do so. Now the trap FS.MDINF is invoked which reads the medium information and returns with the sector information in register D1 and the medium name pointed to by A1. The next step is to have the pointer "dir_name" point to the name of the device. Address register A4 holds the address of the pointer "dir_name", and the loading of this pointer is shown in the operand,()(A6,A1.L),(A4)+.

After the sector information and device name have been received, the sector information being held in register D5 must be moved to register D0. D0 is used to hold the value returned by a function. The registers stored on the stack pointer are now popped from the stack by the command "MOVEM". At this point the program returns to the calling C-Program.

Once returned from the "medinfo()" function, the sector information is returned as a long word with the most-significant-word being the free sectors and the least-significant-word being the total sectors. In order to extract this information from the integer "sectors", two C bit operators are used: the right shift ">>", and the logical AND "&". The device name is pointed to by the pointer "dir_name".

In order to make this program work, the C listing must be compiled. Using the C-68 compiler, this would be a command such as: EX cc;'-c C_file_c', which will stop the compiler after it produces the "C_file_o" file. Now this module is linked with the assembly module "medinfo_o". The command to do this would be: EX ld;'C_file_o medinfo_o'.

After the linking takes place an executable program is produced called "a_out" (the default filename when none is specified). By simply executing "a_out", the program will run, open floppy 1 and print out the sector and device name.

Depending on the assembler used, there may be other directives required to make this assembly module linkable into a C_Program. The Metacomco Assembler was used for this example, but other assemblers should work provided you adjust for their special directives.

QL ROM-CARD

In our endeavor to bring you interesting developments in the QL world, we've discovered an ideal expansion board for those of you with unexpanded machines, or non TRUMP CARD RAM expansion and diskdrive interface.

The ROM-CARD utilizes the expansion port of the QL, so it won't work with the GOLD CARD or TRUMP CARD installed.

Basically, the QL ROM-CARD may be used for the expansion of RAM or ROM, or for the battery protected MOS RAM-DISK. Selection between the different applications is accomplished by positioning a selector switch. Four memory banks are available on the CARD, each with a type selector of it's own, allowing you to choose between installation of a 32K EPROM (27256) for ROM, or 32K STATIC RAM (SRAM)(61256) for RAM/ROM, or the MOS RAM-DISK.

The battery-backup is integrated on the board with automatic recharging. The write-protection switch allows (for the first time), the use of SRAMs (much faster), for easy replacement of EPROMS. The SRAMs don't require an EPROM burner, but instead are programmed directly from the computer using LBYTES and POKE commands. The buffered SRAMs retain their memory contents after power-off, for a period of up to six months before the QL has to be powered up for a few hours to automatically recharge the batteries.

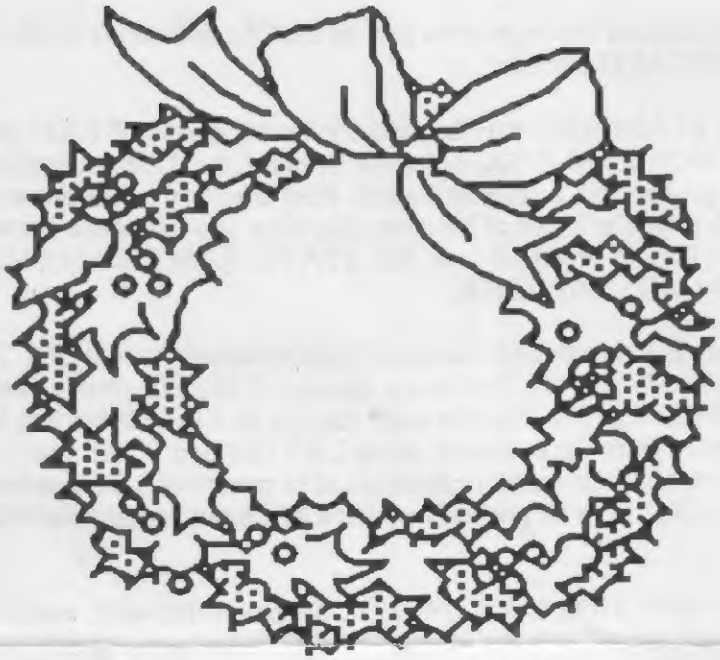
With the addition of the MOS DISK-DRIVER, the power-fail-safe, reset-safe, and write protectable MOS RAM-DISK is the ideal device for users working only with microdrives. You install your frequently used programs and/or data files ONE TIME to the MOS RAM-DISK, they can then be started from the MOS RAM DISK in less than a second, whenever you need them. The RAM used for a program (over 60K for Quill) will remain free for data space.

The QL ROM-CARD is another product of outstanding quality and value, produced by COMPUTER TECHNIK (Jurgen Falkenberg) of Germany and is offered in three configurations:

ROM CARD 128K 0-Wait RAM/ROM, SRAMs/EPROMs, Battery	157 DM
ROM CARD 128K 0-Wait RAM/ROM, with MOS RAM-DISK	175 DM
ROM CARD 128K 0-Wait RAM/ROM, with MOS RAM-DISK & SRAMs	210 DM

There is also a 30 DM postal charge per order (not per item). Order your QL ROM-CARD from:

Computer Technik (Jurgen Falkenberg)
Thanweg 36
D-7539 Ersinden
Germany Tel: 07231 81058



BEST WISHES
THIS
HOLIDAY SEASON
FROM
BOB & THE STAFF